

penMesh—Monte Carlo Radiation Transport Simulation in a Triangle Mesh Geometry

Andreu Badal*, Iacovos Kyprianou, Diem Phuc Banh, Aldo Badano, and Josep Sempau

Abstract—We have developed a general-purpose Monte Carlo simulation code, called *penMesh*, that combines the accuracy of the radiation transport physics subroutines from PENELOPE and the flexibility of a geometry based on triangle meshes. While the geometric models implemented in most general-purpose codes—such as PENELOPE's quadric geometry—impose some limitations in the shape of the objects that can be simulated, triangle meshes can be used to describe any free-form (arbitrary) object. Triangle meshes are extensively used in computer-aided design and computer graphics. We took advantage of the sophisticated tools already developed in these fields, such as an octree structure and an efficient ray-triangle intersection algorithm, to significantly accelerate the triangle mesh ray-tracing. A detailed description of the new simulation code and its ray-tracing algorithm is provided in this paper. Furthermore, we show how it can be readily used in medical imaging applications thanks to the detailed anatomical phantoms already available. In particular, we present a whole body radiography simulation using a triangulated version of the anthropomorphic NCAT phantom. An example simulation of scatter fraction measurements using a standardized abdomen and lumbar spine phantom, and a benchmark of the triangle mesh and quadric geometries in the ray-tracing of a mathematical breast model, are also presented to show some of the capabilities of *penMesh*.

Index Terms—Computer-aided design, Monte Carlo, NCAT, PENELOPE, *penMesh*, triangle mesh.

I. INTRODUCTION

COMPUTER simulations are a valuable tool to study the uses of ionizing radiation. Currently, the most accurate codes for particle transport simulation are those based on

Monte Carlo (MC) methods, and general-purpose codes, such as PENELOPE [1], EGSnrc [2], or GEANT4 [3], are extensively used in a wide range of applications.

A typical MC code is composed of two independent parts: the physics modeling and the geometry tracking. The physics part uses known probability distributions (derived from the atomic cross sections) to sample the distance between interactions with the medium, the energy loss and angular deflection in the interactions, and the generation of secondary particles. The geometry part, also called ray-tracer, handles the transport of particles across the simulated objects and determines the material in which particles move.

The implemented geometric model is one of the most important features of a MC code because it limits the type of applications that can be handled. Most general-purpose codes employ geometries based on the combination of simple primitive objects, such as quadric surfaces (planes, cylinders, ellipsoids, etc.). These primitives are convenient to simulate simple or idealized objects, but they are not well suited to model objects that have free-form (i.e., arbitrary) shapes, such as many organic structures. For this reason some codes have been adapted to use a voxelized geometry (e.g., DOSXYZnrc [4] for EGSnrc, GATE [5] for GEANT4, or *penVox* [6], [7] for PENELOPE), which facilitates the development of anatomical phantoms from segmented computed tomography (CT) scans. A shortcoming of voxelized phantoms is that they have a limited resolution (determined by the voxel size) and are not flexible, that is, the objects inside the voxelized phantom can not be individually translated or deformed. Another problem of current codes is that the implemented geometry models are not standardized. This means that the geometry files can not be used in different codes and have to be created and visualized using tools specifically developed for each code.

To overcome some of these limitations we have developed *penMesh*, a general-purpose MC code based on the PENELOPE subroutine package that employs a standard computer-aided design (CAD) geometry model: triangle meshes. Triangle meshes can approximate any arbitrary surface and, therefore, can represent the boundary of virtually any object. *PenMesh* has been successfully employed in the past to study two medical imaging applications: coronary angiography [8], [9] and prostate brachytherapy imaging [10]. Other MC codes have also been extended to CAD geometries, such as GEANT [11] or EGS4 [12]. A triangle mesh geometry package for PENELOPE had been previously developed [13] but it is not publicly available and it had an extremely low simulation efficiency due to the lack of a spatial data structure to sort the triangles during

Manuscript received January 13, 2009; revised April 10, 2009. First published May 08, 2009; current version published November 25, 2009. This work was supported in part by the U.S. Food and Drug Administration (FDA) Office of Womens' Health, in part by the NIH-NIBIB, and in part by an appointment to the Research Participation Program at the Center for Devices and Radiological Health administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and the FDA. The work of J. Sempau was supported by the Spanish *Ministerio de Educación y Ciencia* under Project FIS2006-07016.

*A. Badal is with the Institut de Tècniques Energètiques, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain, and also with the NIBIB/CDRH Laboratory for the Assessment of Medical Imaging Systems, U.S. Food and Drug Administration, Silver Spring, MD 20993 USA (e-mail: andreu.badal-soler@fda.hhs.gov).

I. Kyprianou, D. P. Banh, and A. Badano are with the NIBIB/CDRH Laboratory for the Assessment of Medical Imaging Systems, U.S. Food and Drug Administration, Silver Spring, MD 20993 USA (e-mail: iacovos.kyprianou@fda.hhs.gov).

J. Sempau is with the Institut de Tècniques Energètiques, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain, and the Networking Research Center on Bioengineering, Biomaterials and Nanomedicine (CIBER-BBN), 08028 Barcelona, Spain.

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TMI.2009.2021615

the ray-tracing. The *penMesh* code and its documentation will be freely distributed.

In this paper, we present a detailed description of the *penMesh* code (Section II) and its triangle mesh ray-tracer algorithm (Section III); three example simulations of transmission X-ray imaging are presented in Section IV; and, finally, the code features, performance, and future applications are discussed in Section V.¹

II. PENMESH CODE

The simulation code *penMesh* [7]–[10] implements a general-purpose MC algorithm that simulates the transport of electrons, positrons and photons in a geometry composed of homogeneous bodies limited by triangle meshes. The interaction between matter and radiation, in the energy range from 50 to 1 GeV, is simulated by the state-of-the-art physics subroutines from the PENELOPE 2006 package [1].

PenMesh uses a hybrid geometry model that combines the simplicity of objects defined by quadric surfaces and the geometric detail of objects represented by triangles. The standard PENELOPE quadric geometry package, PENGEO, is used to define a bounding box that encloses the triangle meshes and to track the particles moving outside this box. The tracking inside the box is handled by a new set of subroutines implementing the ray-tracing algorithm described in Section III-B.

The *penMesh* source code is based on the *penEasy* package [6], a modular general-purpose main program for PENELOPE. Most of the features of *penEasy* are made available to *penMesh*, such as its flexible source models and tallies for the most common quantities of interest (e.g., 3-D dose distributions, energy fluence spectra, energy deposition pulse height spectra, etc.). Taking advantage of the main program modular structure, a new tally for the creation of radiographic images has been developed and implemented in *penMesh*. Due to the wide energy range in which PENELOPE can be applied, the new tally can cover from low-energy X-ray systems to radiotherapy portal imaging. A description of the implemented image formation model and an example simulation are given in Section IV-B.

The *penMesh* main program and the new object-oriented geometry package are coded in C++. However, they make use of the original *penEasy* and PENELOPE subroutines and common blocks in FORTRAN 77. Therefore, the code takes advantages of the advanced features of C++ and the computational efficiency and simplicity of FORTRAN 77. A header file is used to harmonize the naming conventions between the two languages for the GNU Compiler Collection (<http://gcc.gnu.org>) and compatible compilers (for example, the Intel compiler for Linux); *make* scripts are provided to simplify the multilanguage compilation.

MC simulations may require large computing times to produce results with acceptable statistical uncertainties. To overcome this limitation, *penMesh* simulations can be readily executed in parallel in multiple computers using the scripts from the *clonEasy* package [7], [14].

¹The mention of commercial products herein is not to be construed as either an actual or implied endorsement of such products by the U.S. Department of Health and Human Services.

III. TRIANGLE MESH GEOMETRY

PenMesh uses a boundary representation geometry model in which the shape of each object is described by a triangle mesh. The implemented algorithm is not limited to any specific application and imposes only two restrictions on the meshes: being watertight closed (a particle may not enter or exit the mesh without crossing a triangle) and bearing no self-intersecting manifolds (triangles from the same mesh can not intersect, i.e., a particle may not enter the mesh from within itself). The algorithm can handle intersecting meshes from different objects, meshes contained inside other meshes, and overlapping coplanar triangles.

Some advantages of triangle meshes compared to other mathematical surfaces—such as quadrics, nonuniform rational B-splines (NURBS), or Bezier patches—are that triangles can be fitted to any arbitrary surface, can be exactly ray-traced (ray-tracing is costly for cubic or higher-order parametric surfaces), and can be easily generated, manipulated, stored and displayed using existing CAD software. An obvious drawback of the triangles is that curved surfaces can only be approximated up to the smallest triangle size used. Nevertheless, the size of the triangles can be adapted to the required level of detail and no artifacts should appear in the simulation due to tessellation inaccuracies.

The mesh corresponding to each object used in a *penMesh* simulation is stored in an ASCII file in *ply* format.² The *ply* file must contain three different sections: a header giving the number of triangles and vertices in the mesh (other properties, such as normals or color, may be included but are not used); a list of the coordinates of each vertex; and a list of the vertices associated to each triangle (adjacent triangles may share two vertices in a closed mesh). The *ply* files can be created and handled by many of the available CAD toolkits, and there are numerous programs that can export/import these files to/from other popular CAD formats.

A. Spatial Data Structures: The Octree

Every time a particle moves across the triangle mesh geometry it is necessary to check if the particle's trajectory intersects any triangle. A typical *penMesh* geometry contains millions of triangles and, therefore, checking the intersection with every triangle at each step would result in an extremely slow computation [13]. This can be avoided by using a spatial data structure to group adjacent triangles inside bounding boxes. Using this technique only those triangles located inside the boxes crossed by the ray will be checked for intersection, therefore the total number of ray-triangle intersection tests will be significantly reduced at the expense of adding ray-bounding box intersection tests. Fortunately, computing the distance from a point to the wall of an axis-aligned box is very fast and does not introduce a significant overhead in the simulation. The data structures most commonly used for ray-tracing are the uniform grid, the octree, and the k-D tree [15].

The *uniform grid* divides space in axis-aligned equal-size boxes (voxels). The grid ray-tracing is straightforward and can

²The *poly* file format, *ply*, is a standard computer file format used to store three-dimensional data, such as polygonal meshes. This format was developed at the Stanford Computer Graphics Laboratory: <http://www-graphics.stanford.edu/data/3Dscanrep>.

be accelerated by accounting for the periodicity of the intersection between the ray and the voxel walls [16]. The drawback of uniform grids is that they are inefficient in heterogeneous geometries, i.e., when there are regions with high density of triangles and regions with low density or without triangles. In this situation, using small voxels significantly increases the required computer memory while forcing the program to calculate many unnecessary intersections with empty voxels in the low density region. On the other hand, if large voxels are employed the code has to calculate many ray-triangle intersections inside the voxels located on the high density region and the code execution is slow.

The *octree* [17] is a hierarchical tree structure that subdivides space in different size boxes (nodes). The first node (root) is an axis-aligned parallelepiped enclosing all the triangles. This node is divided in eight octants of the same size, which are recursively subdivided in the same way. This subdivision is applied until all space is partitioned according to a predetermined termination condition. The index of subdivision is called the octree level and the final nodes are called leaves. Constructing and ray-tracing an octree is fast and simple because the size and position of each node depends only on its level and on the shape of the root node.

The *k-D tree* [15] is a binary space-partitioning hierarchical structure where space is recursively divided in two parts using axis-aligned planes which alternate the dividing axis in the k dimensions (e.g., alternating planes along the x and y axis in 2-D). The plane position is calculated according to a predetermined splitting condition (e.g., the resulting volumes may contain exactly half of the triangles in the original volume). The uniform grid and the octree are particular cases of a k -D tree with simple splitting conditions. Therefore a k -D tree is theoretically the most efficient spatial data structure when the splitting condition is correctly optimized for the given application and geometry. Nevertheless, the creation and ray-tracing of a general k -D tree is not as simple as for the octree or the uniform grid, and the resulting computer code may require a longer execution time.

Based on the expected efficiency and low overhead, an octree structure was implemented in penMesh. The octree is generated using an heuristic termination condition that subdivides a node whenever the amount of triangles it contains is larger than or equal to the value of the node level. This restrictive condition favors the generation of higher level nodes tightly fit on the object surface and it has exhibited optimum computing performance in our benchmark tests. The maximum octree level can be chosen by the user depending on the complexity of the geometry and the available RAM memory in the computer where the simulation will be executed. Fig. 1 shows a sample level 9 octree sorting the triangles from the anthropomorphic phantom described in Section IV-B. The represented plane contains triangles corresponding to the skin, lungs, heart, liver, ribs, sternum, and a vertebra meshes.

B. Ray-Tracing Algorithm

The new geometric routines implement a robust and exact algorithm to track the movement of particles across triangles contained inside octree leaves.

Ray-tracing inside the triangle mesh region begins by locating the particle position in the octree structure. This is done by searching recursively from the root node to the subnodes

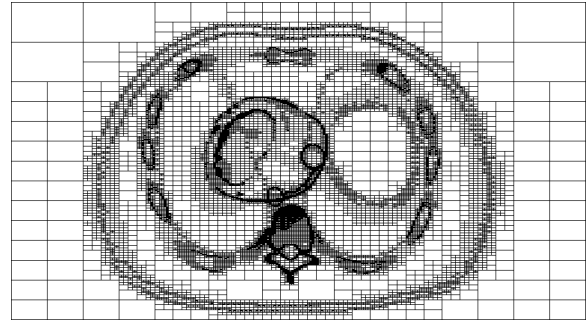


Fig. 1. Node structure of a level 9 octree sorting the triangles from a realistic anthropomorphic phantom.

until the octree leaf that contains the particle position is found. Finding the subnode that contains the particle is trivial because the splitting planes are located exactly at the middle of the node. Then, PENELOPE's physics routines sample the distance to the next interaction using a pseudo-random number generator and taking into account the medium interaction cross sections. The ray-tracer takes the sampled distance and searches for possible interface crossings in the rectilinear path. If the interaction point is found in the current node and no triangle is intersected, the interaction is processed. If the trajectory crosses a triangle, the particle is stopped on its surface and the medium material is updated. This triangle is not checked for intersection in the following leap to avoid getting trapped on the triangle surface due to computer rounding errors. Finally, if the particle reaches the leaf wall it is stopped and the neighbor leaf is loaded. The leap continues in the new leaf without re-sampling the interaction distance, because octree walls are virtual boundaries that merely sort the triangles and do not represent real material interfaces.

A variety of techniques can be used to find the neighboring nodes and transport the particles from one leaf to another [15]. A top-down approach consists of finding the next leaf through a search from the root node. A bottom-up approach moves up to the leaf parent and grandparent nodes and then goes down looking for the neighboring node. The method that we have implemented uses pointers to the six leaf neighboring nodes (with the same or lower level) precalculated during the initialization stage using the top-down approach.

A general and robust ray-tracing algorithm has to address two complicated situations: the intersection of meshes from different objects and the presence of overlapping coplanar triangles. These situations may represent the real geometry or may have been artificially produced during the phantom generation or surface tessellation. Obviously, the geometry must be thoroughly inspected to assure that unrealistic object overlapping does not affect the simulation outcome.

PenMesh deals with triangle mesh intersections by storing the surfaces that are crossed during the particle track in a virtual particle *flight log*. Every time a particle crosses a surface the log is searched for the index of the corresponding object. If the surface was not previously crossed, the particle enters a new object and its index is recorded in the log. If the index is found in the log, the particle is exiting the object and its index is deleted from the log. In the regions where multiple objects overlap the flight log

contains more than one entry of object indices. In such cases, the appropriate material at the current position is determined by using a lookup table that establishes an object priority hierarchy (see [9] and [18, Appendix A]).

The flight log is essential to determine the material found at any point inside the geometry, and the material that will be found when the particle exits the current object. For this reason the generated secondary particles inherit a copy of the flight log at the point they are created. The radiation source may be located inside the triangle mesh region, but in this situation the source routine must be adapted to load the appropriate flight log for each emitted particle. The log can be automatically calculated by transporting a virtual ray from any point outside the triangle bounding box back to the source location.

A particle originating outside the octree region that enters and crosses the whole octree will end up having only one entry in the flight log. This index corresponds to the quadric body that defines the octree bounding box. Having more than one index in the log when leaving the octree indicates that there is an inconsistency in the geometry. Object indices may remain on the flight log if their triangle meshes are not correctly closed or they extend outside the defined bounding box.

The second complicated situation mentioned above is the handling of overlapping coplanar triangles. During the ray-tracing process, particles are stopped on the surface of the intersected triangles. In case the distance to the next triangle is zero—as for coplanar triangles—a particle can get trapped, i.e., continuously jumping between the coplanar triangles without changing its actual location. In a digital computer the intersection distance is calculated with a finite precision and a null distance may be represented by a value close, but not equal, to zero. In this situation the particle could continue the track but the flight log would be corrupted and the particle would move in an incorrect material. To avoid this problem the function that looks for triangle intersections in *penMesh* has been adapted to detect coplanar triangles, defined as those for which the intersection distance is smaller than 10^{-9} cm. All coplanar triangles are crossed at the same time and the corresponding object indices are recorded in the flight log.

The most time-consuming part of the triangle mesh ray-tracing is the calculation of ray-triangle intersections (about 30% of the total simulation time). *PenMesh* calculates these intersections using the efficient algorithm developed by Möller and Trumbore [19]. The intersections with the octree node walls also take a significant amount of execution time (about 15%). Since the nodes are aligned with the axis the ray-box intersection can be found with a simple and fast calculation. For a particle located at (x_p, y_p, z_p) inside a box and moving with a directional vector (u_p, v_p, w_p) with $u_p > 0$, the distance d_x to the nearest box wall perpendicular to the x axis is

$$d_x = \frac{x_{\max} - x_p}{u_p} \quad (1)$$

where x_{\max} is the maximum value of x inside the box. Equivalent equations can be used to find the intersection with the y and z walls. The triangle-box intersections that have to be calculated to distribute the triangles into the leaves during the octree generation are computed using an efficient algorithm developed by Akenine-Möller [20].

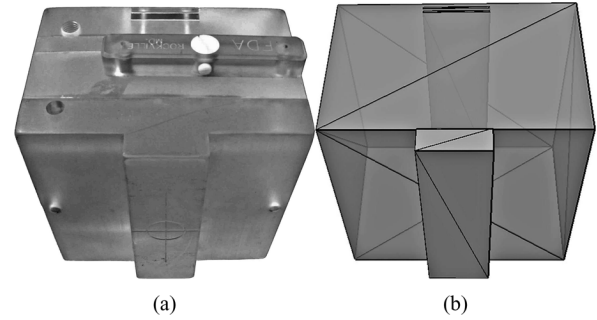


Fig. 2. CDRH abdomen and lumbar spine phantom [27] used in the scatter fraction measurements: (a) picture of the real phantom used in the lab and (b) triangle mesh version containing four objects with 48 triangles in total.

Note that, since ray-tracing the octree structure is much faster than calculating ray-triangle intersections, increasing the resolution of the geometry (i.e., using smaller triangles) will not significantly increase the execution time, provided that the octree level is increased to keep the same average number of triangles per octree leave (for most particle tracks, the number of surfaces crossed will be the same regardless of the resolution).

IV. EXAMPLE SIMULATIONS

The performance of a new MC code is typically evaluated by simulating experiments that can be reproduced in the laboratory or with previously well-established simulation codes. *PenMesh* uses the original physics routines from *PENELOPE* and the MC algorithm from *penEasy*, which have been extensively used and validated with experimental data in the past [21]–[25]. Therefore the primary aim of the example simulations that are presented in this section is to assess the correctness and accuracy of the new geometric routines, rather than to validate the implemented physics models. These examples also highlight relevant features and future applications of *penMesh*.

A. Scatter Fraction Measurements

Scatter is an important factor of image quality degradation in transmission X-ray imaging. For this reason, an accurate simulation of scattered radiation is essential for the simulation of imaging systems. Scatter fraction (SF) measurements also provide a simple way to compare simulated results with experimental data, and were used in the past to validate a CAD geometry package for *EGS4* [26]. We simulated and measured SFs using an idealized model of the human abdomen and lumbar spine developed at the Center for Devices and Radiological Health (CDRH) [27]. Fig. 2 shows the physical phantom used in the lab and the triangle mesh version used by *penMesh*, containing four objects with 48 triangles.³

The SF of a given image acquisition is defined as

$$\text{SF} = \frac{S_s}{S_s + S_p} \quad (2)$$

where S_s is the signal produced by the scattered X-rays and S_p is the signal produced by the primary (nonscattered) X-rays, as recorded in a standard energy integrating X-ray detector [28].

³The triangulated phantom was created with *ParaView*, an open-source, multi-platform application designed to visualize and manipulate digital data. *ParaView* is freely available at the website <http://www.paraview.org>.

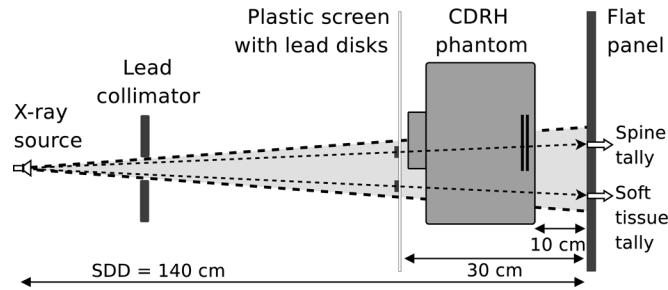


Fig. 3. Experimental setup for the scatter fraction measurement (not to scale). The arrows mark the places where the fractions were tallied. The simulation geometry did not include the lead disks.

The SF can be experimentally measured by inserting a lead disk between the X-ray source and the imaged object. The signal detected at the center of the projected disk is generated by the radiation scattered within the object, and the SF can be measured by dividing this signal (scatter) and the signal detected at the same location without the disk (scatter plus primaries). The diameter of the disk affects the measured SF; however, the value corresponding to a zero-diameter-disk can be estimated by taking multiple measurements for disks with decreasing diameters and extrapolating with a linear fit [28], [29].

Measuring the SF in a MC simulation is straightforward because the scattered and primary particles can be differentiated without using beam blockers. In penMesh, the particles are labeled according to the interactions that they have suffered in the track. Particles are sorted in four different groups: nonscattered (primaries), single elastic scattered, single inelastic scattered, and multiple scattered. Using this information, and assuming that the signal detected in the X-ray detector is proportional to the total energy of the particles entering the scintillator [28], the SF can be directly measured with (2).

A diagram of the experimental setup used to measure the SF is given in Fig. 3. A 90 kVp X-ray source, with 10×10 and 20×20 cm² fields and a source-detector distance (SDD) of 140 cm, was used. The SF was tallied 10 cm downstream the phantom, at two points located behind the soft tissue region (containing 16.91 cm of lucite) and behind the spine region (composed of 0.46 cm of aluminum and 18.95 cm of lucite). The detector used in the experimental setup was a digital flat panel, model Varian 4030CB (Varian Corp. Salt Lake City, UT, USA), with 2048×1596 195×195 μm^2 pixels and a 600- μm -thick columnar CsI(Tl) scintillator. The X-ray tube was a Varian B180 (Varian Corporation, Salt Lake City, UT) with 0.3 mm nominal focal spot and a tube filtration of 1 mm aluminum. The experimental uncertainty in the SF measurement was estimated as 8%.

The simulation used a point source emitting a realistic 90 kVp energy spectrum computed with the software provided by the IPEM Report 78 [30]. The square fields were collimated using a completely absorbing lead collimator defined with quadric surfaces. The SF was estimated by scoring the number of primary and scattered particles that entered a 1.0 cm² sensitive area centered at the point that corresponds to the center of the projected disks in the experimental setup. The lead disks were not included in the simulation geometry because the SF was directly tallied counting the particles that arrived at the sensitive area. The number of initial X-rays for the 10×10 and 20×20 cm² fields

TABLE I
EXPERIMENTAL AND SIMULATED SCATTER FRACTIONS BEHIND TWO REGIONS OF THE CDRH PHANTOM (UNCERTAINTY GIVEN AS 2σ)

Field	Region	Experiment	Simulation	Difference
10×10	soft	0.28 ($\pm 8\%$)	0.26 ($\pm 3\%$)	-7 %
	spine	0.41 ($\pm 8\%$)	0.40 ($\pm 4\%$)	-2 %
20×20	soft	0.49 ($\pm 8\%$)	0.50 ($\pm 4\%$)	2 %
	spine	0.63 ($\pm 8\%$)	0.66 ($\pm 5\%$)	5 %

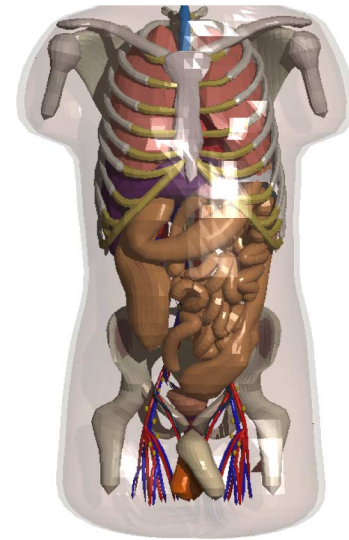


Fig. 4. Tessellated version of the NCAT phantom [18], composed of 330 closed triangle meshes and more than 1.5 million triangles.

were $3 \cdot 10^9$ and $9 \cdot 10^9$, respectively. The simulation speeds, in an Intel Core 2 processor at 2.4 GHz, were 37 127 X-rays/s for the first field and 32 974 X-rays/s for the second.

Table I summarizes the SFs experimentally measured and simulated with penMesh. Using the MC simulation it is also possible to obtain information about the scatter that can not be measured in a real experiment, such as the fraction of particles scattered by each interaction process, or which parts of the geometry contribute most to the detected SF.

B. Simulating a Detailed Anthropomorphic Phantom

PenMesh has been used in the past to simulate two medical imaging procedures [8]–[10]. These simulations were performed with the anthropomorphic phantom shown in Fig. 4, which is composed of 330 closed meshes and more than 1.5 million triangles. A higher resolution version of the phantom, with more than 5 million triangles, is also available. This phantom was derived from the NURBS-based cardiac-torso (NCAT) phantom [18], which is based on the Visible Human Project data (<http://www.nlm.nih.gov/research/visible>); the included heart and coronary arteries were segmented from a high-resolution CT angiography [9], [31]. A comprehensive review of computational anthropomorphic models that could be adapted to penMesh is provided in [32].

PenMesh can generate radiographic images with two different detector models. In the first model the image is formed by tallying the energy deposited inside the detector pixels and dividing by the pixel area. The resulting pixel values have units of eV/cm². In case the detector material is defined as completely radiopaque,

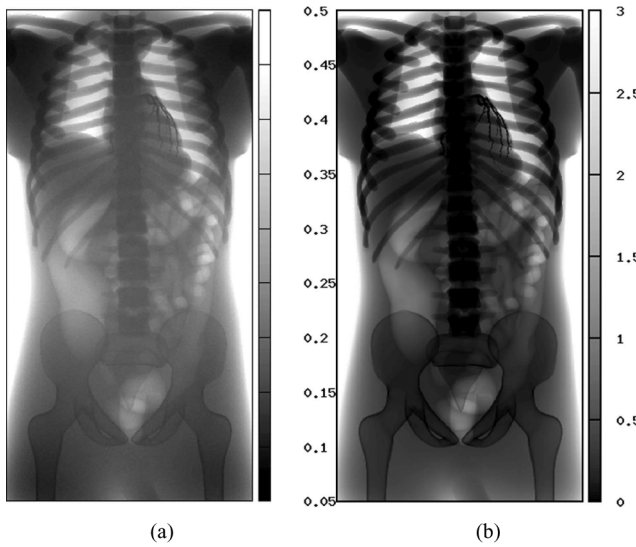


Fig. 5. Whole body radiography with the tessellated version of the NCAT phantom. (a) Monte Carlo simulated image (grey scale: eV/cm^2 per initial X-ray) and (b) fast ray-tracing projection (grey scale: detection probability in %).

the pixel values correspond to the energy fluence on the detector surface. The second model consists in a direct ray-tracing of the geometry. In this case, the simulation is deterministic, not MC. Each pixel value is determined by transporting an X-ray directly from the point focal spot to the pixel center and tallying the probability that the X-ray does not interact in the total path. The interaction probability is calculated using the attenuation coefficients from the PENELOPE database, for the mean energy of the X-ray spectrum. The image created with this idealized model is noise-free, does not have statistical uncertainty, and does not include scattered radiation. Therefore, this model only provides qualitative information about the imaging system, but it is a convenient way to check the geometry before a long MC simulation. A typical simulation requires only a few minutes of execution time with the ray-tracing model, while the accurate MC simulation may require several days.

A whole body radiography of the tessellated NCAT phantom was simulated to show some of the capabilities of *penMesh* and test the robustness of its new geometry package. Fig. 5 presents the images obtained using the two aforementioned image formation models. The simulations employed a realistic 90 kVp X-ray source [30] and a $50 \times 100 \text{ cm}^2$ detector with $0.1 \times 0.1 \text{ cm}^2$ pixels. The detector material was defined to absorb all the incoming radiation and therefore the detecting efficiency was 100% and there was no scatter inside the detector. The phantom coronary arteries were filled with an iodine contrast agent to facilitate their visualization in the projection image. The full MC simulation, Fig. 5(a), used 10^{11} X-rays and was executed in parallel in 30 computers for approximately four days using the *clonEasy* scripts [14]. A level 8 octree was used in the simulation. For each execution, 386 MB of memory were required and the simulation speed was 14420 X-rays/s per CPU.⁴ The average statistical uncertainty in the pixel values, for pixels with values above half the image maximum value, was below 1%. The ray-tracing simulation, Fig. 5(b), required only 3 min of execution in a single CPU. This

⁴The reported timing results correspond to an Intel Xeon CPU at 2.0 GHz and 4 GByte of RAM. The code was compiled with GCC -O3.

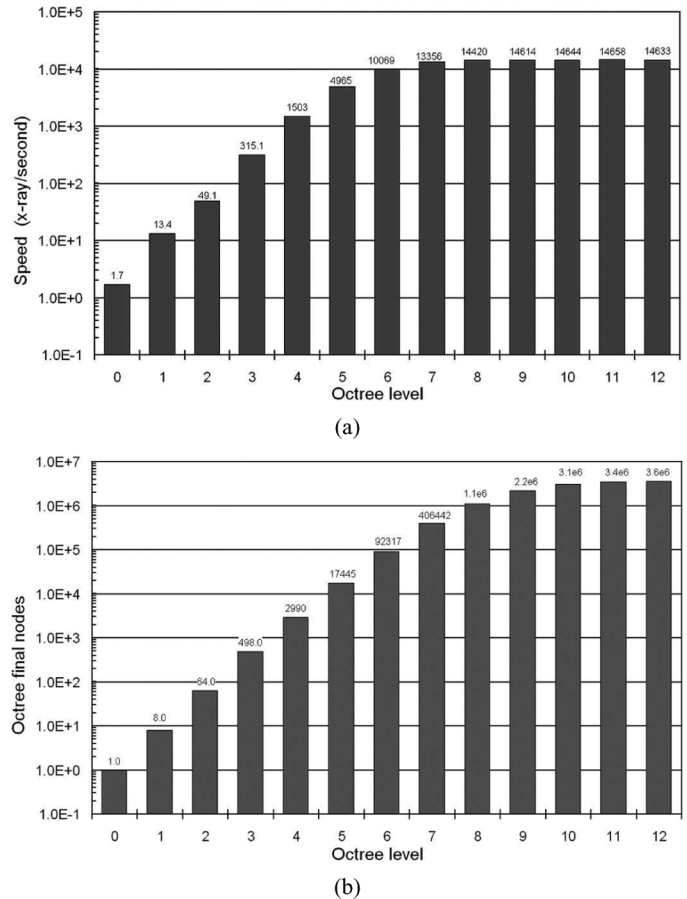


Fig. 6. Simulation performance as a function of the octree level: (a) simulation speed (in X-rays/s per CPU) and (b) number of octree final nodes.

time includes about 2 min of initialization (the time spent reading the triangles, creating the octree structure, reading the material cross section database, etc.).

The effect of the octree level on the performance of *penMesh* was evaluated for this particular application. Fig. 6(a) displays the simulation speed and (b) the number of octree final nodes (leaves), as a function of the octree level.

C. Geometric Accuracy of *penMesh*

The accuracy of the new triangle mesh geometry routines was evaluated by comparing the simulation of an idealized mammography system with *penMesh* and with a version of PENELOPE employing its standard quadric geometry. A simple mathematical breast model which, when imaged, produced a circular signal and a randomly structured background was employed [33]. The phantom consisted of a $6 \times 6 \times 5 \text{ cm}^3$ box filled with a material equivalent to adipose tissue (density $0.92 \text{ g}/\text{cm}^3$). The box contained 125 spheres, as shown in Fig. 7(a), with diameters ranging from 1.0 to 0.5 cm and made of mammary gland tissue ($1.06 \text{ g}/\text{cm}^3$). A 1-cm-diameter spherical mass made of a fictitious glandular material 50% more dense than nominal tissue ($1.60 \text{ g}/\text{cm}^3$) was placed at the center of the box. In the standard PENELOPE simulation, the geometry was described using quadric surfaces (i.e., perfect spheres); in *penMesh*, the surface of each sphere was described using 20 000 triangles, as displayed in Fig. 7(b). Due to the inherent inaccuracies of the tessellation

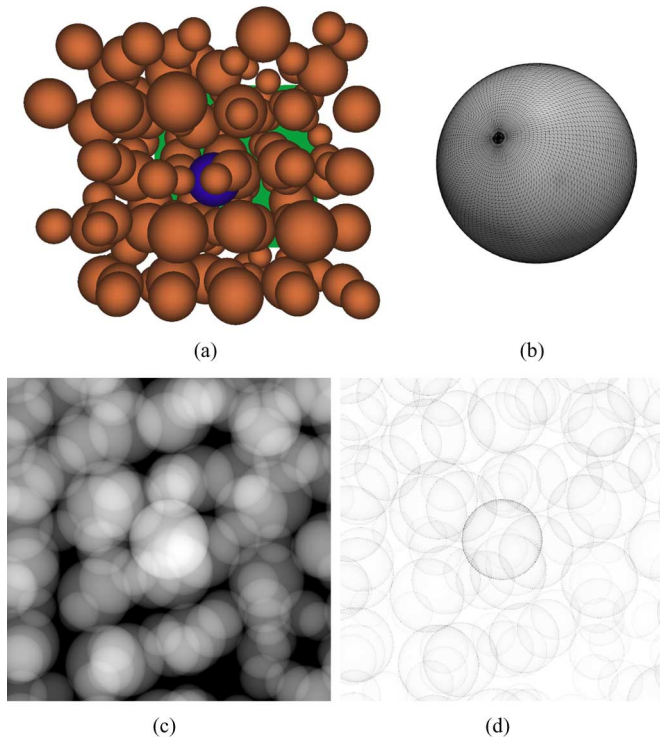


Fig. 7. Comparison of the quadric and the triangle mesh geometries: (a) volume rendering of the simulated phantom and detector, as seen from the source; (b) tessellated sphere used in penMesh, composed of 20 000 triangles; (c) noise-free projection of the quadric phantom; and (d) relative difference between the projections computed with quadrics and with triangles (linear gray scale, maximum difference of 0.83% represented in black).

process, the volume of each mesh was 0.1% smaller than the corresponding sphere (the triangle vertices were located on the surface of the spheres). The phantom was irradiated with a 20 keV X-ray point source. A $6 \times 6 \text{ cm}^2$ detector with $100 \times 100 \mu\text{m}^2$ pixels was defined 70 cm downstream the source.

Accurate MC simulation was used to estimate the dose deposited in the central sphere for the two geometry models. After a 12-h-long simulation, the tallied dose with the quadrics was $71.1 \pm 0.4 \text{ eV/g}$ per X-ray and the simulation speed was 746 X-rays/s; with the triangle meshes, the dose was $71.0 \pm 0.2 \text{ eV/g}$ per X-ray and the speed was 3975 X-rays/s. Projection images were also obtained by ray-tracing the phantoms using the idealized ray-tracing model described in the previous section. Fig. 7(c) displays the noise-free projection obtained using quadrics, and Fig. 7(d) presents the relative difference between this image and the projection obtained using triangles. The maximum relative difference found in the pixel values was 0.83%, the mean difference 0.03%, and the median difference 0.02%.

V. DISCUSSION

In the previous section, we have shown that the new simulation code can robustly handle complex triangle mesh geometries, and that the use of parallel execution allows the obtention of results with low statistical uncertainty in an affordable time. These two key features allow penMesh to simulate some applications that can not be currently studied with state-of-the-art general-purpose MC codes. As an example, Fig. 5 shows how

penMesh can successfully simulate medical imaging systems with a complex CAD phantom, and that the resulting images exhibit clinically-realistic features.

The octree spatial data structure is an essential part of penMesh. Fig. 6(a) shows that increasing the octree level results in an exponential acceleration of the simulation, until a certain level for which the acceleration saturates. The saturation takes place when most of the space is completely sorted by the octree and, therefore, it depends on the defined meshes and the implemented termination condition. The speed without the octree (level 0) was 1.7 X-rays/s; the speed increased to 4965 X-rays/s with a level 5 octree and up to 14 420 X-rays/s for level 8: an 8483-fold acceleration. Above level 8 the acceleration saturated and the speed for level 11 was 14 658 X-rays/s, only 1.6% faster than for level 8. For level 12, the speed was even less than for level 11, showing that the time spent ray-tracing the new nodes was not compensated by a reduction in the triangle intersection tests. The number of octree leaves also increases exponentially with the octree level, as seen in Fig. 6(b). However, the number of leaves saturates at a higher level than the simulation speed, indicating that the leaves created above certain level are not relevant to the overall simulation.

The study of the SF is relevant in imaging applications because scattered radiation significantly reduces the contrast and the signal-to-noise ratio in the radiographic images. The agreement between the measured and simulated fractions was relatively good, taking into account the large experimental (8%) and simulation (3%–5%) uncertainties. The differences of up to 7% were expected because the techniques used for the measurements were different: beam blockers were used in the lab, while the simulation calculated the scatter fraction directly labeling the particles that had been scattered. In addition, it was assumed that there was no transmission of radiation through the lead disks and no scatter coming from the collimator. Furthermore, the scatter fraction was experimentally tallied in the nine pixels closer to the center of the projected disks (sensitive area of $3.6 \cdot 10^{-3} \text{ cm}^2$) while—for simulation efficiency reasons—the sensitive area in the simulation was much bigger (1 cm^2). Other sources of discrepancy were the idealized models used to simulate the radiation source (point focal spot, nominal energy spectrum, etc.) and the detector (signal proportional to the deposited energy, 100% detection efficiency, no scattering in the scintillator, no optical photon transport or energy dependent optical photon generation, etc.).

In Section IV-C, the new triangle geometry and the standard quadric geometry were compared. The agreement in the projection images was excellent; as expected, the pixels at the edge of the sphere projections had a small difference (below 1%) caused by the approximate way the sphere was tessellated. The tallied doses in the central sphere for both geometries were in good agreement (0.1% difference). In case the triangle geometry was incorrectly implemented, the particle trajectories would be significantly different in the two geometries and the tallied dose would be different.

Through the example simulations we showed three different methods to generate a triangle mesh geometry: adapting an existing CAD model (such as the tessellated NCAT phantom shown in Fig. 4), segmenting CT images (such as the heart and

coronary arteries seen in Fig. 5), or using specialized CAD software [such as the phantom shown in Fig. 2(b)]. The ability to use quadric surfaces to define objects outside the triangle mesh region was useful to model objects that can be easily described using planes and cylinders, such as the radiation source, the beam collimator or the flat-panel detector. For simple objects, such as Fig. 2, the speed of the simulation using triangles or quadrics is similar because in both cases more than half of the execution time is spent by the physics subroutines. For complex objects, the triangle mesh geometry is typically faster than the quadrics due to the efficient space partitioning of the octree data structure; as an example, the simulation of the phantom shown in Fig. 7(a) was five times faster with triangles than with quadrics.

VI. CONCLUSION

The new MC simulation code *penMesh* has been described in detail, and it has been demonstrated that this code can accurately simulate complex objects described by triangle meshes. We expect that the possibility to use advanced CAD software to create the geometry for a general-purpose PENELOPE simulation will simplify the development of sophisticated geometries and the study of very realistic situations. As an example, we have shown that *penMesh* can be readily applied to the study of some medical imaging applications, including those requiring detailed anatomical phantoms.

REFERENCES

- [1] F. Salvat, J. M. Fernández-Varea, and J. Sempau, *PENELOPE—A code system for Monte Carlo simulation of electron and photon transport*. Issy-les-Moulineaux, France: Nuclear Energy Agency (OECD), 2006.
- [2] I. Kawrakow and D. W. O. Rogers, The EGSnrc code system: Monte Carlo simulation of electron and photon transport Nat. Res. Council Canada, Tech. Rep. PIRS-701, 2006.
- [3] S. Agostinelli *et al.*, “Geant4—A simulation toolkit,” *Nucl. Instrum. Meth. Phys. Res. A*, vol. 506, pp. 250–303, 2003.
- [4] B. Walters, I. Kawrakow, and D. W. O. Rogers, DOSXYZnrc Users Manual Nat. Res. Council Canada, Tech. Rep. PIRS-794, 2007.
- [5] S. Jan *et al.*, “GATE: A simulation toolkit for PET and SPECT,” *Phys. Med. Biol.*, vol. 49, pp. 4543–4561, 2004.
- [6] J. Sempau and A. Badal, “penEasy—A generic, modular main program and voxelised geometry package for PENELOPE,” *Nucl. Instrum. Meth. Phys. Res. B. Code* [Online]. Available: <http://www.upc.es/intel/downloads/penEasy.htm>, submitted for publication
- [7] A. Badal, “Development of advanced geometric models and acceleration techniques for Monte Carlo simulation in medical physics,” Ph.D. dissertation, Univ. Politècnica de Catalunya, Catalunya, Spain, 2008 [Online]. Available: <http://www.tdx.cat/TDX-0523108-095624>
- [8] A. Badal, I. Kyprianou, A. Badano, J. Sempau, and K. J. Myers, J. Hsieh and M. J. Flynn, Eds., “Monte Carlo package for simulating radiographic images of realistic anthropomorphic phantoms described by triangle meshes,” in *Proc. Med. Imag. 2007: Phys. Med. Imag.*, 2007, vol. 6510.
- [9] I. S. Kyprianou, A. Badal, A. Badano, D. P. Banh, M. Freed, and K. J. M. Thompson, K. R. Cleary and M. I. Miga, Eds., “Monte Carlo simulated coronary angiograms of realistic anatomy and pathology models,” in *Proc. Med. Imag. 2007: Visualizat. Image-Guided Procedures*, 2007, vol. 6509.
- [10] A. Badal, I. Kyprianou, A. Badano, and J. Sempau, “Monte Carlo simulation of a realistic anatomical phantom described by triangle meshes: application to prostate brachytherapy imaging,” *Radiother. Oncol.*, vol. 86, pp. 99–103, 2008.
- [11] J. Sulkimo and J. Vuoskoski, “Particle tracking in sophisticated CAD models for simulation purposes,” *Nucl. Instrum. Meth. Phys. Res. A*, vol. 371, pp. 434–438, 1996.
- [12] J. Tabary and A. Glière, “Coupling photon Monte Carlo simulation and CAD software. Application to x-ray nondestructive evaluation,” in *MONTE CARLO 2000*, Lisbon, Portugal, 2000.
- [13] N. Borglund, J. Eriksson, E. Fumero, P. Kjäll, L. Mårtensson, C. Orsholm, and T. Sikora, “Geometry package for Monte Carlo simulations on CAD files,” *Nucl. Instrum. Meth. Phys. Res. A*, vol. 525, pp. 417–420, 2004.
- [14] A. Badal and J. Sempau, “A package of Linux scripts for the parallelization of Monte Carlo simulations,” *Comput. Phys. Commun.*, vol. 175, pp. 440–450, 2006.
- [15] A. Chang, A survey of geometric data structures for ray tracing Polytechnic Univ., New York, Tech. Rep. TR-CIS-2001-06, 2001.
- [16] J. Amanatides and A. Woo, “A fast voxel traversal algorithm for ray tracing,” *Eurographics*, vol. 87, pp. 3–10, 1987.
- [17] A. Glassner, “Space subdivision for fast ray tracing,” *IEEE Comput. Graph. Applicat.*, vol. 4, pp. 15–22, Oct. 1984.
- [18] W. P. Segars, “Development of a new dynamic NURBS-based cardiac-torso (NCAT) phantom,” Ph.D., Univ. North Carolina, Raleigh, 2001.
- [19] T. Möller and B. Trumbore, “Fast, minimum storage ray-triangle intersection,” *J. Graph. Tools*, vol. 2, pp. 21–28, 1997.
- [20] T. Akenine-Möller, “Fast 3D triangle-box overlap testing,” *J. Graph. Tools*, vol. 6, pp. 29–33, 2001.
- [21] J. Sempau, J. M. Fernández-Varea, E. Acosta, and F. Salvat, “Experimental benchmarks of the Monte Carlo code PENELOPE,” *Nucl. Instrum. Meth. Phys. Res. B*, vol. 207, pp. 107–123, 2003.
- [22] J. Sempau, P. Andreo, J. Aldana, J. Mazurier, and F. Salvat, “Electron beam quality correction factors for plane-parallel ionization chambers: Monte Carlo calculations using the PENELOPE system,” *Phys. Med. Biol.*, vol. 49, pp. 4427–4444, 2004.
- [23] S.-J. Ye, I. A. Brezovich, P. Pareek, and S. A. Naqvi, “Benchmark of PENELOPE code for low-energy photon transport: Dose comparisons with MCNP4 and EGS4,” *Phys. Med. Biol.*, vol. 49, pp. 387–397, 2004.
- [24] A. Cot, J. Sempau, D. Pareto, S. Bullich, J. Pavía, F. C. no, and D. Ros, “Study of the point spread function (PSF) for ^{123}I SPECT imaging using Monte Carlo simulation,” *Phys. Med. Biol.*, vol. 49, no. 14, pp. 3125–3136, 2004.
- [25] M. Vilches, S. Garcia-Pareja, R. Guerrero, M. Anguiano, and A. M. Lallena, “Monte Carlo simulation of the electron transport through thin slabs: A comparative study of PENELOPE, GEANT3, GEANT4, EGSnrc and MCNPX,” *Nucl. Instrum. Meth. Phys. Res. B*, vol. 254, pp. 219–230, 2007.
- [26] F. Mathy, R. Guillemaud, and M. Picone, “Experimental validation of a coupled photon Monte Carlo and CAD software,” *Nucl. Instrum. Meth. Phys. Res. A*, vol. 504, pp. 317–320, 2003.
- [27] Standardized Methods for Measuring Diagnostic X-Ray Exposures American Institute of Physics Tech. Rep., 1990.
- [28] H. P. Chan and K. Doi, “The validity of Monte Carlo simulation in studies of scattered radiation in diagnostic radiology,” *Phys. Med. Biol.*, vol. 28, pp. 109–129, 1983.
- [29] I. Brezovich and G. Barnes, “A new type of grid,” *Med. Phys.*, vol. 4, pp. 451–453, 1977.
- [30] K. Cranley, B. J. Gilmore, G. W. A. Fogarty, and L. Desponds, Catalogue of diagnostic X-ray spectra and other data Inst. Phys. Eng. Med., Tech. Rep. 78, 1997.
- [31] D. P. Banh, I. S. Kyprianou, S. Paquerault, and K. J. Myers, J. P. W. Pluim and J. M. Reinhardt, Eds., “Morphology-based three-dimensional segmentation of coronary artery tree from CTA scans,” in *Proc. Med. Imag. 2007: Image Process.*, 2007, vol. 6512.
- [32] H. Zaidi and X. G. Xu, “Computational anthropomorphic models of the human anatomy: The path to realistic Monte Carlo modeling in radiological sciences,” *Annu. Rev. Biomed. Eng.*, vol. 9, pp. 1.1–1.30, 2007.
- [33] S. Park, R. Leimbach, H. Liu, I. Kyprianou, R. Jennings, A. Badano, and K. J. Myers, E. Samei and J. Hsieh, Eds., “A task-based evaluation method for X-ray breast imaging systems using variable background phantoms,” in *Proc. Med. Imag. 2009: Phys. Med. Imag.*, 2009, vol. 7258.